Regression on House Dataset

By Salih Eren Yüzbaşıoğlu

AI Club AR-GE

REGRESSION EXPLANATION

Regression is a statistical technique used to find the relation between one dependent variable and one or more other independent variables. We would like to select independent variables such that they reduce sum of squares.

EXPLORING MODELS

1. Linear Regression

Imagine you're trying to predict something, like how well you'll do on a math test based on the number of hours you study and the amount of sleep you get. Multiple Linear Regression helps us understand how multiple factors (variables) influence an outcome we're interested in. In simpler terms, it's like trying to find the best equation that connects several things to predict something else.

Equation:

In Multiple Linear Regression, we use a equation to make predictions. It looks something like this:

$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n$

Let's break it down step by step:

Y represents the outcome we want to predict, like our math test score.

b_0 is called the intercept. It's like a starting point when all other variables are zero. In our example, it could be the score you'd get if you didn't study or sleep at all.

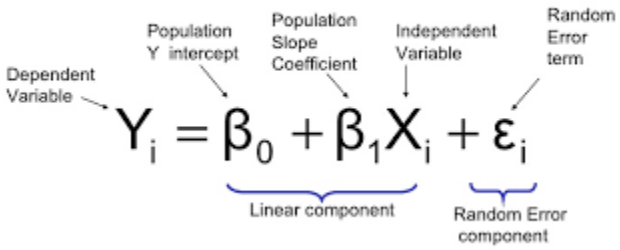
$b_1, b_2, ..., b_n$ are called the coefficients. These numbers tell us how much each variable ($X_1, X_2, ..., X_n$) contributes to the outcome. We'll find the best values for these coefficients later.

$X_1, X_2, ..., X_n$ represent the variables we think might affect the outcome. In our example, X_1 could be the number of hours you study, and X_2 could be the hours of sleep you get.

We need to find the best values for the coefficients ($b_0, b_1, b_2, ..., b_n$). We do this using a technique called "least squares." It's like trying different combinations of coefficients to minimize the difference between our predicted outcomes and the actual values we have.

Once we find the best coefficients, we can plug them back into the equation we discussed earlier. This equation becomes our prediction model! We can now use it to estimate the outcome (Y) based on the values of the variables ($X_1, X_2, ..., X_n$) we have.

Below figure represents a general equation for linear regression:



2. KNN Regression

K-Nearest Neighbors (KNN) Regression, a clever technique that harnesses the collective wisdom of neighboring data points to make predictions. Imagine you have a bunch of data points with known outcomes, and you want to predict the outcome for a new data point. KNN Regression helps us find the most similar data points (neighbors) and uses their outcomes to estimate the outcome for the new data point.

The idea behind KNN Regression is pretty straightforward. We assume that similar data points should have similar outcomes. So, when we want to predict the outcome for a new data point, we look at its K nearest neighbors (K being a user-defined value) in the feature space. These neighbors are determined based on the similarity of their features to the features of the new data point.

To find the K nearest neighbors, we measure the distance between the new data point and all the other data points in the dataset. The most common distance metric used is Euclidean distance, which you might remember from your math class. It calculates the straight-line distance between two points in the feature space.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Once we have the distances, we select the K data points with the smallest distances to the new data point. These become the neighbors that will contribute to our prediction.

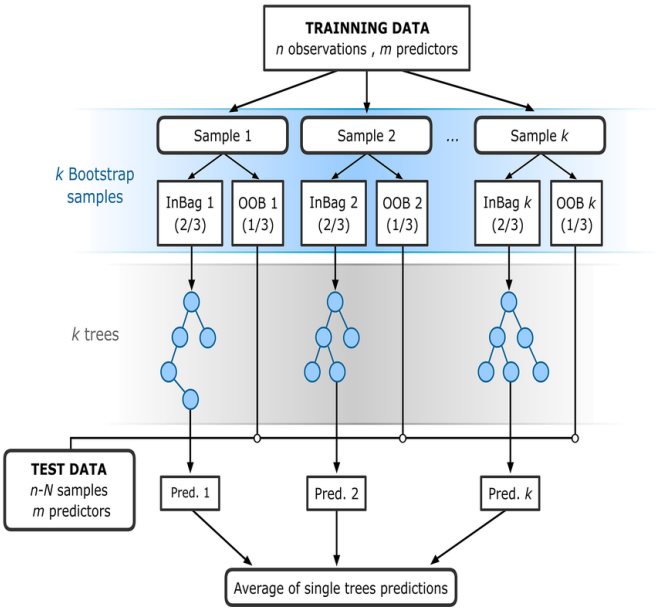
Now that we have our K nearest neighbors, we need to estimate the outcome for the new data point. In KNN Regression, we take a weighted average of the outcomes of the neighboring data points. The weights are determined by the inverse of the distances between the new data point and its neighbors.

Closer neighbors have a higher influence on the prediction, while farther neighbors have a lower influence. For example, if we're predicting the price of a house based on its size and number of bedrooms, we would look at the K nearest neighbors (houses) and calculate their weighted average price. This average price becomes our prediction for the new house.

The choice of K is crucial in KNN Regression. A smaller K value (e.g., 3) leads to predictions that are more influenced by local fluctuations, potentially resulting in a high variance. On the other hand, a larger K value (e.g., 10) considers more neighbors and may produce smoother, but potentially biased, predictions. Selecting the right value of K requires some experimentation and understanding of the dataset and problem at hand

3. Random Forest Regression

Random Forest Regression combines the strength of multiple decision trees to create a robust and accurate prediction model. It's like having a group of experts working together to provide you with the best possible prediction. But before we dive into Random Forest Regression, let's understand the concept of ensemble learning. Ensemble learning is all about combining the predictions of multiple models to make a more accurate and reliable prediction. In the case of Random Forest Regression, the models are decision trees. Imagine a decision tree as a flowchart with a series of "yes" or "no" questions leading to a final prediction. Each question is based on a feature or attribute of the data, and the branches represent different outcomes or predictions. Decision trees are great at capturing complex relationships between features and outcomes.



Now, let's see how Random Forest Regression takes advantage of ensemble learning to make predictions. Here's how it works:

Random Forest Regression constructs a bunch of decision trees, each trained on a different subset of the training data. These trees are created using a process called bootstrapping, where random samples of the original data are selected with replacement. This technique ensures diversity among the trees.

To introduce more randomness and prevent overfitting, Random Forest Regression selects a random subset of features at each split of the decision tree. This means that each tree only considers a subset of the available features, improving the model's generalization capability.

Once all the decision trees are built, they work together to make predictions. When a new data point arrives, each tree in the random forest independently predicts an outcome. In the case of regression, these predictions are numerical values. The final prediction is then obtained by taking the average (or majority vote) of all the individual tree predictions

4. SVM Regression

SVM Regression helps us find the optimal line (or hyperplane) that best fits our data by maximizing the margin between the data points and the line. It's like finding the perfect balance between simplicity and accuracy.

Before we delve into SVM Regression, let's understand the concept of Support Vector Machines. SVM is a powerful algorithm used for both classification and regression tasks. In the case of regression, it aims to find a line (or hyperplane) that best fits the data points, while also minimizing the prediction errors.

In SVM Regression, the margin refers to the distance between the line and the closest data points. The objective is to find the line that maximizes this margin. The data points that lie exactly on the margin or within a certain distance from it are called support vectors. These support vectors play a crucial role in defining the regression line.

To find the optimal regression line, SVM Regression uses a mathematical optimization technique. The algorithm searches for the line that minimizes the prediction errors while

maintaining the maximum possible margin. This process involves solving a quadratic optimization problem, but don't worry, the SVM algorithm takes care of the complex math for us.

One of the key features of SVM Regression is the kernel trick. It allows us to transform the data into a higher-dimensional feature space, where the regression problem may become easier to solve. The kernel function calculates the similarity between data points in this transformed space, enabling us to capture more complex relationships.

SVM Regression offers various types of kernels to choose from, depending on the characteristics of the data. Some commonly used kernels include:

Linear Kernel: Used for linear relationships between features and outcomes.

Polynomial Kernel: Suitable for capturing non-linear relationships with polynomial functions.

Radial Basis Function (RBF) Kernel: Effective for capturing non-linear and complex relationships.

Choosing the appropriate kernel involves experimentation and understanding the data and problem at hand.

5. Neural Network Regression

Neural networks consist of interconnected layers of artificial neurons (also known as nodes or units). Each neuron takes input, performs a computation, and passes the output to the next layer. In regression tasks, the output layer usually consists of a single neuron that provides the predicted numerical value.

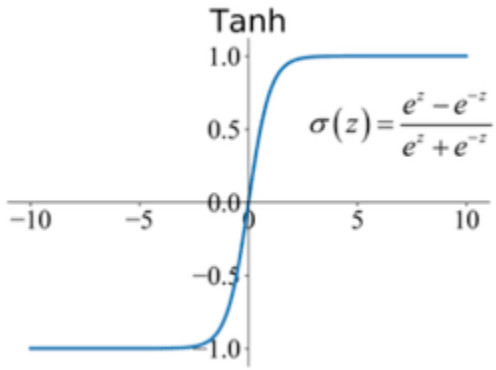
Activation functions play a crucial role in neural networks. They introduce non-linearities, enabling the network to learn complex relationships between features and outcomes. Popular activation functions for regression include the Rectified Linear Unit (ReLU) and the hyperbolic tangent (tanh) function. I used the tanh function in my implementation.

Training a Neural Network:

To train a neural network for regression, we need a labeled dataset containing input features and corresponding target values. The network learns by adjusting its internal parameters (weights and biases) through a process called backpropagation. Backpropagation involves iteratively calculating the gradient of the loss function with respect to the network parameters and updating them using optimization algorithms like Stochastic Gradient Descent (SGD).

Loss Functions:

Loss functions measure the discrepancy between predicted values and true target values. In regression tasks, common loss functions include Mean Squared Error (MSE), Mean Absolute Error (MAE), and Huber loss. These functions quantify the difference between predicted and actual values, providing feedback for the network to adjust its parameters.



6. Gradient Regression

Gradient Boosting Regression, also known as Gradient Boosted Trees, is a popular ensemble learning method that sequentially builds a strong predictive model by iteratively improving upon the weaknesses of previous models.

Decision trees are flowchart-like structures that make predictions by recursively splitting the data based on different features until reaching a final prediction. However, individual decision trees tend to have limitations, such as high variance or bias, which can affect their predictive performance.

Gradient Boosting Regression overcomes the limitations of individual decision trees by sequentially building a strong model. Here's how it works:

Gradient Boosting Regression starts by building a simple base model, often a decision tree, to make an initial prediction. This base model is trained using the available training data.

In each iteration, Gradient Boosting Regression builds a new decision tree that predicts the errors (residuals) of the previous model. The goal is to reduce these residuals and improve the overall prediction. The new tree is then added to the ensemble, and its predictions are combined with the predictions of the previous models.

To determine the direction and magnitude of the improvements, Gradient Boosting Regression uses gradient descent optimization. It calculates the gradient of the loss function with respect to the predicted values and adjusts the predictions of the new tree accordingly. This process ensures that subsequent models focus on the areas where previous models performed poorly.

The final prediction is obtained by combining the predictions of all the individual models in the ensemble. Typically, the predictions are weighted averages, where the weights are determined during the training process.

Data Preprocessing

After reviewing the data, I identified a significant number of unique values in the Address column. To mitigate potential overfitting and labeling issues during regression, I made the decision to remove this column. I then proceeded to check for null values, and fortunately, none were found, which was advantageous for the analysis.

Among the remaining columns, there were only two categorical variables: City and Province. Given the requirement to maintain the order of the data, I recognized potential challenges in utilizing one-hot encoding. This method could lead to certain values consistently being false during training, as they would not appear in the test data. To address this concern, I opted to replace both the City and Province columns with the average house price of their respective cities/provinces. This approach ensured that the categorical information was retained while avoiding the encoding issue.

Subsequently, I performed outlier removal by excluding values that deviated more than three standard deviations from the mean. This step aimed to eliminate any extreme data points that could potentially skew the analysis.

Finally, I normalized all columns to have a mean of 0 and a standard deviation of 1. This normalization process standardizes the data, enabling fairer comparisons and reducing the influence of individual column scales on the analysis.

Evaluation and Conclusion

Model	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	R-Squared
Multiple Linear Regression	0.46	0.42	0.52
kNN Regression	0.53	0.45	0.46
Random Forest Regression	0.38	0.38	0.61
Support Vector Regression	0.49	0.43	0.50
Neural Network Regression	0.40	0.39	0.60
Gradient Regression	0.43	0.40	0.57
EXTRA: My Own Neural Network Implementation	0.33	0.41	0.63

First and foremost, it is important to note that the Extra model I developed was solely trained and tested on the first 100 rows of data. Consequently, its values should not be considered for comparison with other models.

Upon examining the regression results in the table, it is evident that Random Forest Regression consistently outperforms the other models in terms of several evaluation metrics. Specifically, it exhibits the lowest Mean Squared Error (MSE), the lowest Mean Absolute Error (MAE), and the highest R-squared value.

Random Forest Regression emerges as the clear winner across all evaluation criteria, while the Neural Network model demonstrates similar performance, albeit slightly inferior. The Gradient model follows closely behind. Interestingly, the KNN regression model performed the poorest, achieving an R-squared value of 0.46.

Regarding the remaining models, we can deduce that their performance falls within the average range.